DISTRIBUTED & SCALABLE DATA ENGINEERING

Zeeshan Ahmed Lodhia

Project Report

Geo-location Clustering using the k-means Algorithm

Introduction:

Loud acre Mobile a fictional wireless service has all the service across the USA. In this project we are trying to find out the similarities among the group of customers, for that we are taking the help of amazon webservices S3 bucket, RDD and EMR cluster etc. and we are trying to visualize with the help of geographical location i.e., latitude and longitude. Here we are part from using the AWS services for data engineering we are trying to use k-means cluster and we are trying to understand the market in locations by clustering and grouping them, we are trying to understand the needs of customer by grouping them together.

Tools and Technologies:

Spark EMR cluster, a S3 bucket, RDD (Resilient Distributed Data Base). Those are the requirements for the project.

With the help of functional programming, and matplotlib and seaborn and other visualization libraries we try to get a visualize output on the data. For clustering we are using K-means model.

Approach:

For storing all our data, we have created S3 buckets, and upload all the data files in the S3 buckets. Later the preprocessing is done on the data, as there are three files, and with multiple delimiters, in order to import to get the data in a clean readable format the data multiple delimiters are used. After analyzing the data we have to understand what datatype each and every data is of, then later after understanding we are mapping each data points with its relative datatype using the map(). As the data types of latitude and longitude has decimal values it has been mapped with a float datatype. Now there are numerous values with “0” aka null values no we have to handle those data. So, as we are focusing more on clustering people on the basis of latitude and longitude, so

filtering is done on the data. Now we have the processed data reading for the analytics. There is a function called as SaveAsTextFile() this function comes in handy, where we can use this processed data and we can save it in the S3 bucket. So, we can use for further computation. Later that spark data frame is convert into pandas data frame for visualization purpose using toppandas() method.

Now we try to analyze the data with the help of data visualization using matplotlib and seaborn. Seaborn and matplotlib are like state-of-the-art libraries for data visualization. Now we try to analyze device status file and with the help of matplotlib we try to plot the graph of the latitude of longitude, which is shown in fig 1, after we plot the graph of lat long text file, and it's shown in fig 2. Later we try to plot sample_geo text file and its graph is shown in fig 3.

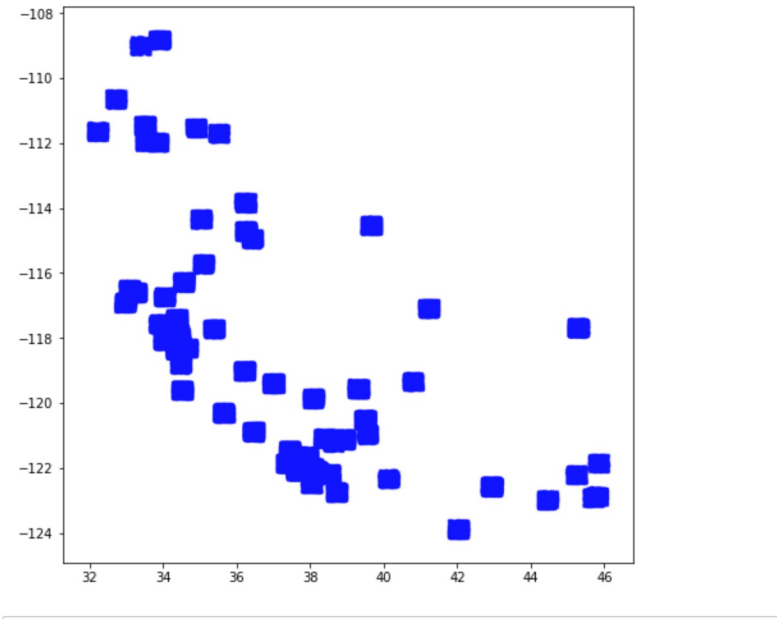


Figure 1 : Latitude vs longitude on status_file

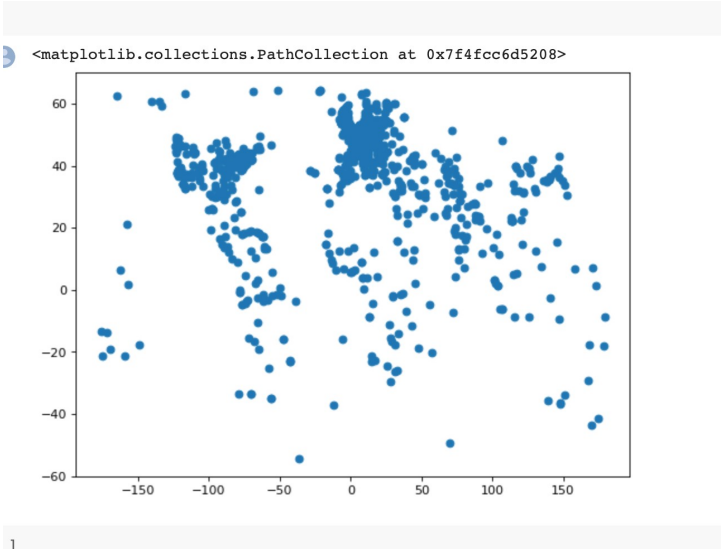


Figure 2 : Graph of latitude vs longitude on lat long text file

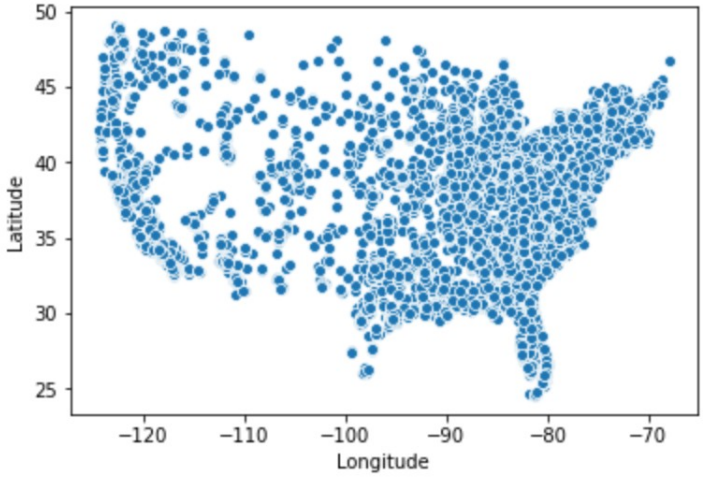


Figure 3 : Graph of Latitude vs longitude on sample_geo text file

After plotting a graph, we try to build k-means model by taking the help of [1]. In every iteration, every single data point is assigned to the nearest cluster based on a distance metric, it could be Euclidian distance or great circle distance. Now as we are dealing with geo location, so the best metric for the distance is great circle distance. K-means groups the data by minimizing the sum of squared distance between the data points and their closest centroid. Using [1]. K means model for data here the numbers of clusters has been used is ‘2,4,5,6’ following is the output for three models using multiple clusters.

```
Silhouette with squared euclidean distance = 0.7779851895575357
Cluster Centers:
[  38.02864791 -121.23352192]
[  34.29718423 -117.78653245]
[  43.98989868 -122.77665336]
[  34.58818551 -112.35533553]
[  42.25924472 -116.90267328]
--- 12.97334909439087 seconds ---
```

Now we calculate the Euclidean and great circle distance for the points. With the help of [2],[3],[4], calculation of the Euclidean and great circle distance between points has been done.

original_latitude	original_longitude	prediction	center_latitude	center_longitude	gc_dist	eu_dist
33.689476	-117.543304	1	34.297184	-117.78653	35.59862841593989	0.42846743379777763
37.43211	-121.48503	0	38.02865	-121.23352	34.96130983668151	0.41911582615284715
39.43789	-120.93898	0	38.02865	-121.23352	79.38456955250766	2.0727134647313505

only showing top 3 rows

Following are the output on the visualization of the different clusters, Fig. 4, Fig 5, Fig 6, Fig 7, Fig 8, are visualization of the clusters.

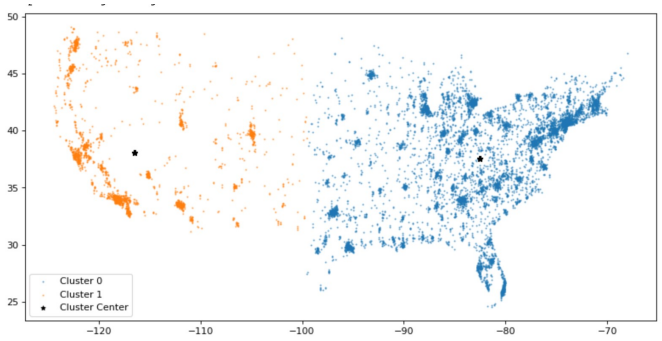


Figure 4: Location data for K = 2, 4

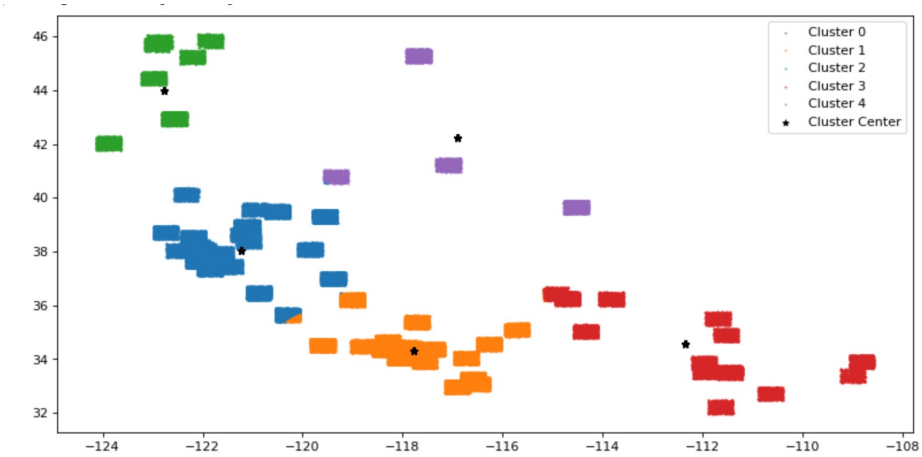


Figure 5 : Different Cluster output

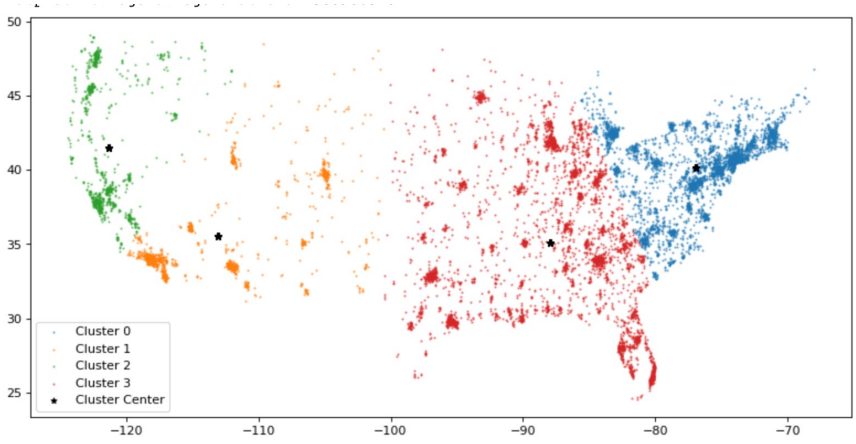


Figure 6 : Location data when K = 5

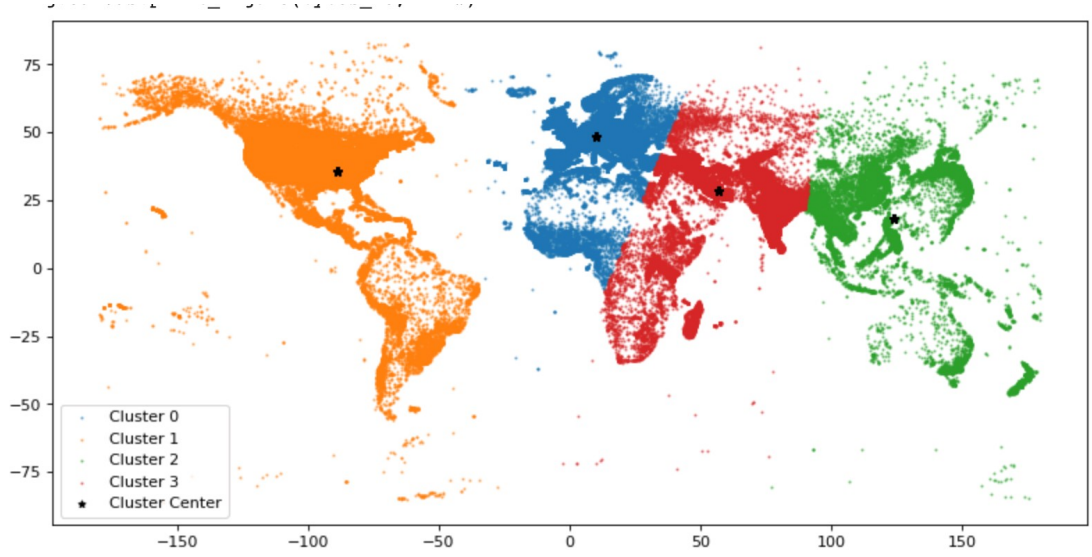
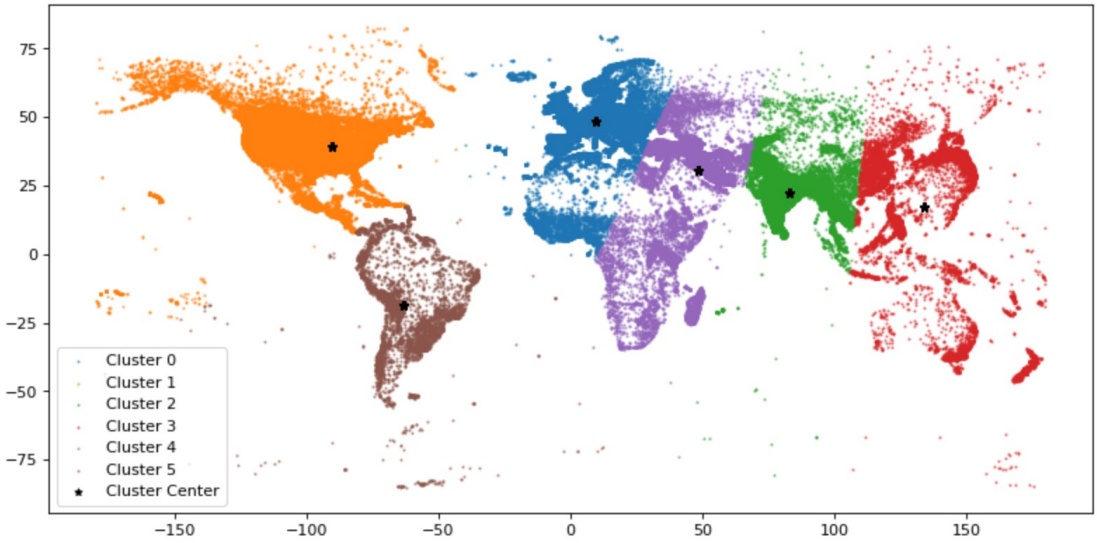


Fig 7 : Location data when K = 4



Now after the clustering is done, we do some runtime analysis first one is the one which is on EMR cluster, where is the other one is one with a local mode with two threads, we try to find the performance and runtime analysis as we try to compare which performs better, following is the graph of the runtime analysis.

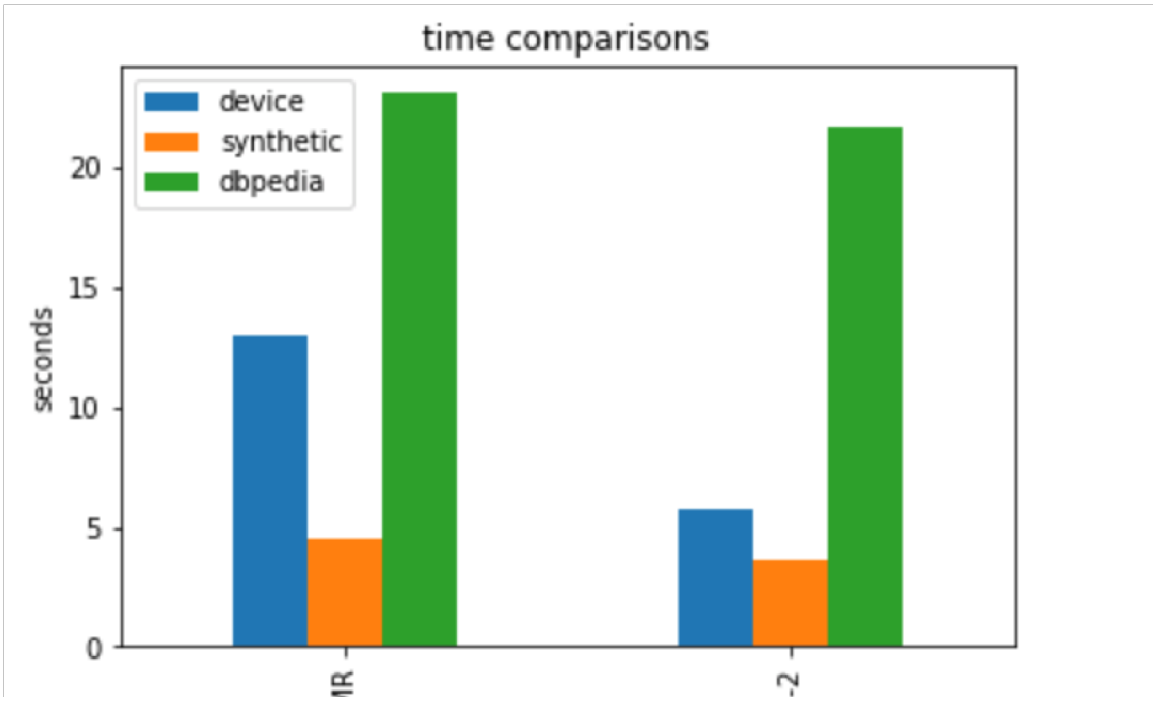


Figure 9: Runtime Analysis

Final Conclusion

After doing the analysis on the visual representation of the above data, we come to know from all the above files we see the data like sites, they have listed the sites and we can understand from that is what the user is interested, and we can group them accordingly. We can take the help of the analysis and assist the industries to focus on the needs to improve business. Pattern extraction and recognition is done with the help of clustering algorithm, as the data is all about geolocation, if we have some more data regarding the data reach and how the service is at particular place then we can find out which part needs more service, and which place the service should be maintained to keep the hold of the customers, and give them proper service what they want, or maybe even better service.

References

1. https://spark.apache.org/docs/latest/ml-clustering.html

2. https://gist.github.com/pavlov99/bd265be244f8a84e291e96c5656ceb5c

3. https://medium.com/@nikolasbielski/using-a-custom-udf-in-pyspark-to-computehaversine-distances-d877b77b4b18

4. https://stackoverflow.com/questions/4913349/haversine-formula-in-python-bearing-anddistance-between-two-gps-points